Preprint
UCRL-JC-144996

Intelligent Collection Environment for an Interpretation System

W. J. Maurer

This article was submitted to 2001 Berkeley Initiative in Soft Computing International Workshop on Fuzzy Logic and the Internet, Berkeley, CA, August 14 – 17, 2001

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

July 19, 2001

Intelligent Collection Environment for an Interpretation System

William J Maurer
DSP Labs, WJM Inc.
Livermore, California, USA
maurer@dsplabs.com

Abstract

An Intelligent Collection Environment for a data interpretation system is described. The environment accepts two inputs: A data model and a number between 0.0 and 1.0. The data model is as simple as a single word or as complex as a multi-level/multi-dimensional model. The number between 0.0 and 1.0 is a control knob to indicate the user's desire to allow loose matching of the data (things are ambiguous and unknown) versus strict matching of the data (things are precise and known). The environment produces a set of possible interpretations, a set of requirements to further strengthen or to differentiate a particular subset of the possible interpretation from the others, a set of inconsistencies, and a logic map that graphically shows the lines of reasoning used to derive the above output.

The environment is comprised of a knowledge editor, model explorer, expertise server, and the World Wide Web. The *Knowledge Editor* is used by a subject matter expert to define Linguistic Types, Term Sets, detailed explanations, and dynamically created URI's, and to create rule bases using a straight forward hyper matrix representation. The *Model Explorer* allows rapid construction and browsing of multi-level models. A multi-level model is a model whose elements may also be models themselves. The *Expertise Server* is an inference engine used to interpret the data submitted. It incorporates a semantic network knowledge representation, an assumption based truth maintenance system, and a fuzzy logic calculus. It can be extended by employing any classifier (e.g. statistical/neural networks) of complex data types. The *World Wide Web* is an unstructured data space accessed by the URI's supplied as part of the output of the environment.

By recognizing the input data model as a query, the environment serves as a deductive search engine.

Applications include (but are not limited to) interpretation of geophysical phenomena, a navigation aid for very large web sites, monitoring of computer or sensor networks, customer support, trouble shooting, and searching complex digital libraries (e.g. genome libraries).

1. Introduction

In recent years, much effort has gone into studying the problem of computer interpretation of interacting data from multiple sources or sensors [1]. A number of prototype (automated or semi-automated) systems have been developed. Due to the combinatorial explosion of possible data/interpretations, an automated interpretation system prohibits a single algorithm. Automated statistical/neural networks/expert systems have been partially successful. These systems have been beneficial as a good first order filter to classify the data as having obvious or non-obvious interpretations. A limitation of these systems is that when dealing with non-obvious interpretations, the system needs to be integrated better with the user problem solving methods.

Data must be gathered and analyzed in a timely manner to improve the chances of interpretation and not overwhelm or waste the system resources (e.g. battery power). In support of the requirements for accurate and efficient data collection and analysis, the system must be able to draw upon diverse (and potentially limited) data sources. In order to do this thoughtfully, the capability to rank the relative worth of data must be integrated into the system. The value of data in proving or refuting a particular hypothesis and the potential cost in resources to obtain the data must be balanced by the system.

We propose an "Intelligent Collection Environment for an Interpretation System". The environment allows the user to tailor the use of whatever data sources are available at the time of operation. If a data source is unavailable or is not properly configured for a hypothesis, then the system must make do with the data at hand. The system achieves these adaptive data collection and evaluation objectives through the use of "soft computing" techniques. The basic premise underlying soft computing is "Exploit the tolerance for imprecision,

uncertainty and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality".

The problem falls into a class of problems we define as being semi-structured. By semi-structured we mean problems where human judgment is essential but can be improved by using automation tools. In this approach, we show that non-obvious interpretations require a high degree of interactive analysis and adaptation of methods. In addition to interactive analysis, a general capability of analysis of data and transformations on many scales allows the user to exercise problem-solving methods that are appropriate for the issue at hand. Rapid visualization and direct manipulation of the results allows the user to explore the data space and develop interpretations adaptively. The modern web browser used to access the World Wide Web has proved to be an effective tool for the exploration of unstructured data spaces.

2. Semi-structured problems: structure and approach

The difference between a problem that is structured and one that is unstructured helps to determine where automation tools are appropriate. A solution to a problem that can be programmed is one for which clear rules and a computer program can be defined. The complex process of production scheduling is best dealt with by a linear programming model. There are very definite rules relating inputs to outputs and production to costs, the problem possesses a fundamental deep underlying structure, too complex for the human mind to easily grasp in its detailed entirety but easy for a computer to resolve. At the same time there are many unstructured problems, an extreme example of one is a romance novel. No amount of formal analysis can solve the dilemma.

An unstructured problem does not permit the programming of a solution. The objectives, trade-offs , relevant information, and methodology for analysis cannot be predetermined. Some problems are unstructured simply because of a lack of knowledge or an unwillingness to explore the problem in depth. The degree of potential structure in a problem predefines the procedures, types of computation and analysis, and the information to be used. In a highly unstructured problem, the user must rely on personal judgement, often especially in identifying just what the problem is. There are many differences in the design of a system to support unstructured problems as compared with structured ones. Most obviously, the activities of the user are more central for a system to support unstructured problems. The user initiates and controls the problem solving process and sequence, and uses judgment, personalized objectives, and interpretations to guide the choice of solution. In structured problems, the system will be designed and most of the effort of building the system will be put into the development of routines and sequences of analysis designed to GIVE answers.

3. Browsing in a semi-structured problem domain

A semi-structured problem domain often requires a search for a solution to be found. The solution to be found lies somewhere in the data-space. When the data-space is highly structured, the solution can be retrieved directly by performing some computation. When the data-space is unstructured, the appropriate mechanism for retrieval of a solution is browsing. Browsing is exploratory searching which assumes little knowledge about the structure of the domain being searched [2]. Browsing is available in two different styles.

Navigation is an iterative process in which a user examines the neighborhood of a solution, picks a solution from this neighborhood, examines its neighborhood, and so on.

Probing is a mode in which solutions computed are either a hit (i.e. acceptable) or a miss. When the solution is a miss it can be viewed as being an over-qualification of a solution. In this mode every miss initiates a set of retractions that attempt to broaden the scope of the solution.

Imagine a customer in a store searching for a particular item. The most efficient method to locate this item is to consult a directory and then access the correct shelf. If the customer cannot describe or does not know the item he is looking for, or if the store is not organized in any meaningful way then the customer must apply a *browsing* search technique. Often browsing is done by strolling down the isles, adjusting direction and speed according to the items encountered and the proximity to the desired item. It may also involve hit-and-miss attempts where a customer goes directly to a shelf where he hopes the desired item will be found.

Navigation is analogous to strolling along the aisles of a store. On the other hand, a user who attempts to compute solutions without sufficient familiarity with

the data-space is like the store browser who makes a hit-and-miss attempt by going directly to a shelf in the store. This is analogous to probing. What characterizes probing is that it will fail frequently. What most customers do is use a combination of probing and navigation. The customer may go to a shelf to use as a starting point for navigation.

3.1 Probing using analysis tools

The operations that may be applied to the model objects are *domain independent* and *domain dependent* in nature.

Domain independent operations are used for analysis in many problem domains. These operations don't compute a hit or miss solution as a probe is defined above to. Instead these operations allow the user to compute features which allow the data space to be segmented into regions of interest. Examples of these types of operations are signal processing algorithms and statistical algorithms.

Domain dependent operations are used for analysis in a single domain. These operations do compute a hit or miss solution as a probe is defined above to. These operations allow the user to either use the solution as a starting point for further navigation or to confirm the validity of a solution obtained through navigation. Examples of these types of operations are:

- Model-free estimators such as neural networks that have been trained using features from a particular domain
    - Model-based classifiers such as Bayesian classifiers that have been developed using statistics gathered
    - Rule-based operators used as interpretation tools to allow the user to incorporate non-deterministic feature interpretation operations
    - fuzzy logic operators are used to compute multiple features with a measure of certainty and/or precision.

4.Results

The current environment prototype consists of an interactive model construction and understanding tool for multi level models called Analyst Assistant (AA). The tool, consisting of the World Wide Web, a Java GUI, and a Lisp based expertise server, takes full advantage of the modern web browser and integrates traditional domain independent analysis methods with intelligent domain specific tools for the exploration and analysis of semi-structured problems.

The GUI interface consists of a Knowledge Editor and Model Explorer. The Knowledge Editor is used to define Linguistic Types, Term Sets, detailed explanations, and dynamically created URI's, and to create rule bases using a straight forward hyper matrix representation The Model Explorer allows rapid construction and browsing of multi-level models. By a multi-level model we mean a model whose elements may also be models themselves. One of the features of the Model Explorer is that it can expand a model element. This feature allows the user to probe and recognize meaningful features in the model elements. Models of sub-models and different granularities can be rapidly selected with the mouse interface and analyzed.

The inference engine used to interpret the model incorporates:

- Semantic Network Knowledge Representation [3]
- Assumption-based Truth Maintenance System [4]
- Fuzzy Logic Calculus [5]

Each fact in the Semantic Network has a fuzzy grade that represents how well the value of the fact represents itself as a member of a fuzzy set. A global user-defined grade called the CROSSOVER-POINT is set to a number between 0.0 and 1.0. The CROSSOVER-POINT represents the transition from false to true and is used in pattern matching as follows:

- BELIEVED facts are grade >= CROSSOVER-POINT
- DISBELIEVED facts are grade < CROSSOVER-POINT
- KNOWN facts are grade >= 0
- UNKNOWN facts do not exist

By adjusting the CROSSOVER-POINT, you are allowed to instruct the inference engine to loose match your model with the rule set used to interpret your model (CROSSOVER-POINT closer to 0) or to strict match your model with the rule set used to

interpret your model (CROSSOVER-POINT closer to 1). Usually you will want to start your interpretation of a model in a loose matching mode (things are ambiguous and unknown). As you gain experience with your model and the real world equivalent you will want to interpret your model in a strict matching mode (things are precise and known).

The inference engine interprets your model by applying a rule set that has been previously defined using the Knowledge Editor. The input to the inference engine is:

- the model data
- the names of the rule sets previously defined using the Knowledge Editor
- the value of the CROSSOVER-POINT

Given the input described above, the output interpretation of the
model is an html page full of hyper links and consists of:

- a set of possible interpretations of the data,
- additional data required to strengthen an interpretation or
  differentiate an interpretation from others in the set of possible interpretations, with data ranked according to value,
  - inconsistencies between an interpretation and the data,
  - a logic map that graphically shows the lines of reasoning being used to derive the above output,

In summary, the tools in the system help the user to perform flexible analysis, generate a tentative set of interpretations with further data requirements, and explore the effects of interpretations at different levels of abstraction.

5. References

[1] William J. Maurer, Farid U. Dowla, "Seismic Event Interpretation Using Fuzzy Logic and Neural Networks", Lawrence Livermore National Laboratory, Livermore CA., UCRL-ID- 116130, January 1994.

[2] Motro, A., Browsing in a Loosely Structured Database, SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984, ACM Press, pp. 197-207

[3] Norman, D., Explorations in Cognition, 1975, WH Freeman and Company, pp. 35-59

[4] Johnson, R. R., T. W. Canales, D. L. Lager, C. L. Mason, and R. M. Searfus. (1987). Interpreting Signals with an Assumption-Based Truth Maintenance System. Proc. SPIE, vol. 786, pp. 332-337.

[5] Zadeh, L., The Concept of a Linguistic Variable and its Application to Approximate Reasoning, Information Sciences, 1975 vol. 9, pp. 199-249.

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.